A Appendix

In this supplemental material, we will include the details of dataset split in Section A.1, the experimental results on base classes in Section A.2, the ablation study of hard and soft assignment in Section A.3, implementation and training details of Prototypical VoteNet in Section A.4, implementation details of the baseline method Meta VoteNet in Section A.5, visualization of basic geometric primitives in Section A.6, KNN baselines in Section A.7, non-updated prototypes in Section A.8, performance on the unbalance Problem in Section A.9 and limitation analysis in Section A.10.

A.1 Dataset Split

Table 1 lists the names of novel classes for FS-SUNRGBD and FS-ScanNet.

FS-SUNRGBD	**FS-ScanNet**(Split-1)	**FS-ScanNet**(Split-2)
table, bed, nightstand, toilet	sofa, window, bookshelf, toilet, bathtub, garbagebin	bed, table, door, counter, desk, showercurtain

Table 1: Names of novel classes for **FS-SUNRGBD** and **FS-ScanNet**.

A.2 Results on Base Classes

Method	**FS-ScanNet**				**FS-SUNRGBD**	
	Split 1		**Split 2**			
	AP_{25}	AP_{50}	AP_{25}	AP_{50}	AP_{25}	AP_{50}
VoteNet+TFA	51.00	24.22	51.50	30.73	36.57	10.61
VoteNet+PT+TFA	52.78	25.46	52.04	30.57	38.03	15.44
Meta VoteNet	52.13	25.13	47.89	28.44	40.22	20.60
VoteNet	57.96	32.60	54.63	35.76	47.77	26.78
Ours	53.83	28.20	51.22	32.41	46.07	25.26

Table 2: Results on **FS-ScanNet** and **FS-SUNRGBD** on base classes using mean Average Precision (mAP) at two different IoU thresholds of 0.25 and 0.50, denoted as AP_{25} and AP_{50}.

Since we detect both base and novel classes during inference, we also report the performance on base classes in Table 2. For simplicity, we average the results of all k-shot (*e.g.* k=1,3,5) in each split. Table 2 shows VoteNet achieves the best performance on base classes, because it is specifically designed as a fully supervised method for base classes. Compared with VoteNet, the performance of VoteNet+TFA and VoteNet+PT+TFA on base classes degrades by -6.96% AP_{25} and -8.38% AP_{50}, and -5.18% AP_{25} and -7.14% AP_{50}, respectively, on split-1 of FS-ScanNet. Moreover, Prototypical VoteNet retains much more ability to recognize base classes, which achieves 53.83% AP_{25} and 28.20% AP_{50} on split-1 of FS-ScanNet, and 46.07% AP_{25} and 25.26% AP_{50} on FS-SUNRGBD. Compared with our method, we can observe a significant performance degradation in finetuning based methods. The reason is that, without the large-scale pre-training, a detector cannot learn more generic feature representation so as to transfer knowledge from base classes to novel classes. Under this circumstance, if we force the knowledge transferring by finetuning, it will distort the learned feature space for base classes.

A.3 Hard vs. Soft Assignment

Method	**3-shot**		**5-shot**	
	AP_{25}	AP_{50}	AP_{25}	AP_{50}
Soft	27.24	12.82	29.13	15.13
Hard	31.25	16.01	32.25	19.52

Table 3: Ablation study of hard and soft assignment.

We leverage the hard assignment in Section 3.2.1: Geometric Prototype Construction, in the main paper. Here, we compare the original hard assignment in our implemented method with the soft assignment, which calculates the similarity between a point feature with all geometric prototypes and updates all geometric prototypes in a soft manner by the similarity scores between a point feature and the geometric prototypes. We conduct the experiment on 3-shot and 5-shot in split-1 of FS-ScanNet. The results in Table 3 indicate that the hard assignment is more effective than the soft assignment. This is because the geometric prototypes by the hard assignment are more distinctive than those using the soft assignment, since one geometric prototype is updated using the nearest point features without considering the others.

A.4 Implementation and Training Details

We follow recent practice [4, 6] to use PointNet++ [5] as a default backbone network. The backbone has 4 set abstraction layers and 2 feature propagation layers. For each set abstraction layer, the input point cloud is sub-sampled to 2048, 1024, 512, and 256 points with the increasing receptive radius of 0.2, 0.4, 0.8, and 1.2, respectively. Then, two feature propagation layers successively up-sample the points to 512 and 1024. Additionally, in both PHM and PVM, the multi-head attention layer are combined with feed forward FC layers. The details can be referred to our code. The size of memory bank of geometric prototypes is set to 120. The number of heads in both multi-head attention networks is empirically set to 4. The momentum coefficient is set to 0.999. We use 40k points in FS-ScanNet and 20k points in FS-SUNRGBD as input and adopt the same data augmentation as in [4], including a random flip, a random rotation, and a random scaling of the point cloud by [0.9, 1.1].

Following the episodic training strategy [7], a training mini-batch in Prototypical VoteNet is comprised of a K-shot R-class support set D_{support} and a R-class query set D_{train} (the classes in D_{support} and D_{train} is consistent). Each sample in D_{support} is the whole point cloud scene. Before the pooling step, we use the groud-truth bounding boxes to obtain point features of target objects. The network is trained from scratch by the AdamW optimizer with 36 epochs. The weight decay is set to 0.01. The initial learning rate is 0.008 in FS-ScanNet and 0.001 in FS-SUNRGBD. Additionally, during the inference stage, we input both the query point cloud and the given support set to Prototypical VoteNet. Note that the features of the support point cloud only need to be extracted once, as they are independent of the query feature extraction.

A.5 Implementation Details of Meta VoteNet

We provide more details of the competitive baseline corresponding to Meta VoteNet in our main paper. It is derived from an effective few-shot 2D object detection approach - Meta RCNN [7]. In Meta RCNN, each RoI feature is fused with R class prototypes using the channel-wise multiplication operator. As a result, a number of R fused RoI features for each RoI are generated. Then each fused RoI feature is fed into the prediction layer for the binary prediction (whether the RoI feature is the category that the class prototype belongs to). More details can be referred to in [7]. We incorporate this meta-learning approach into VoteNet, namely Meta VoteNet. Similarly, after Sampling $\&$ Grouping (see Figure 2 in the main paper), we fuse point features of objects with R class prototypes by the channel-wise multiplication operator. Then the binary class prediction and bounding box regression are conducted based on the fused features for classification and location prediction, respectively. For a fair comparison, Meta VoteNet shares the same architecture with Prototypical VoteNet, except Prototypical Vote Module and Prototypical Prediction Module. The training details follow the proposed Prototypical VoteNet.

A.6 Visualization of Basic Geometric Primitives

In Figure 1, we visualize the relation between the learned geometric prototypes and the 3D points by searching points with features that are similar to a given geometric prototype. First, we feed object point clouds to a trained Prototypical VoteNet. Second, for each point feature, we can search for its most similar prototype. If the similarity is above a threshold, we can assign the point to that prototype. Third, we use a density-based clustering algorithm DBSCAN to cluster the point groups, and we draw the minimum 3D bounding box around each point group. As shown in the figure, all the red bounding boxes within each subfigure belong to the same prototype. The result shows that in each subfigure, the enclosed geometric structures are similar. For example, subfigure (a) illustrates that the prototype learns the feature of corners, while subfigure (b) shows that the prototype learns the long stick.

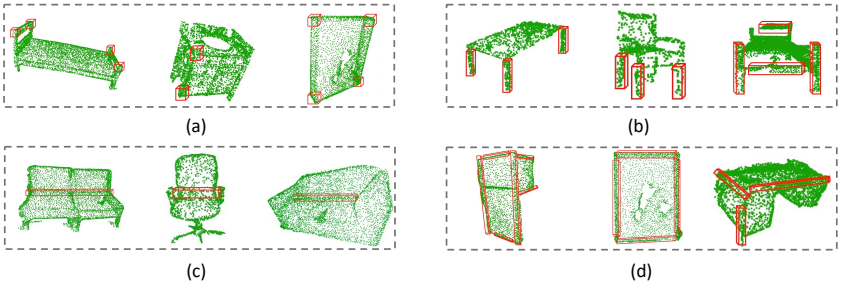

Figure 1: Visualization of some basic geometric primitives learned by the prototypes. (a) Corner, (b) Stick, (c) Hinge, (d) Edge.

A.7 KNN Baselines

Method	3-shot		5-shot	
	AP_{25}	AP_{50}	AP_{25}	AP_{50}
VoteNet + KNN	23.07	9.56	25.58	13.51
Group-Free + KNN	24.22	9.97	26.33	13.92
3DETR[2] + KNN	24.08	10.21	26.01	14.36
Ours	31.25	16.01	32.25	19.52

Table 4: KNN baselines.

We apply KNN assignment to VoteNet and two SOTA 3D detectors GroupFree [2] and 3DETR [3]. We conducted this experiment on 3-shot and 5-shot in split-1 of FS-ScanNet. The KNN assignment is realized by calculating the distance between each object feature and features of all training objects in the classification step, and assigning the sample to the class based on voting from its k-nearest objects of the training set. Here, we take k as one since we find increasing the value k doesn't improve performance. The results are shown in Table 4. Comparing the performance of "VoteNet + KNN" and "ours", we see that the non-parametric KNN classifier will not help improve few-shot learning much ("VoteNet" vs "VoteNet+KNN"). Comparing the performance of different detectors ("VoetNet+KNN", "GroupFree + KNN", and "3DETR+KNN"), we observe that a better detection architecture does not bring large performance gains in the few-shot 3D detection scenario. The most challenging issue for few-shot 3D object detection still lies in how to learn effective representation if only a few training samples are provided. The classifier and architecture don't help much if the model cannot effectively extract features to represent novel categories with only a few samples.

A.8 Non-Updated Prototypes

Method	3-shot		5-shot	
	AP_{25}	AP_{50}	AP_{25}	AP_{50}
Update	28.05	13.89	28.51	14.51
Non-Update	31.25	16.01	32.25	19.52

Table 5: Does setting the prototype at the end (no updates) perform well ?

As shown in Table 5, for the proposed Prototypical VoteNet, if we don't update the prototype in PVM, the performance would degrade significantly. Without updating, the randomly initialized prototypes can not learn the geometry information from base classes in the training phase. In this case, it is hard to transfer the basic geometry information from base classes to the novel classes as the prototypes are meaningless.

Method	P		10P		25P		50P	
	AP_{25}	AP_{50}	AP_{25}	AP_{50}	AP_{25}	AP_{50}	AP_{25}	AP_{50}
VoteNet	62.34	40.82	52.06	35.64	43.12	27.13	40.01	26.77
Ours	62.59	41.25	52.60	36.87	44.53	29.17	41.99	29.01

Table 6: Performance on the Unbalance Problem in ScanNet V2

Method	P		10P		25P		50P	
	AP_{25}	AP_{50}	AP_{25}	AP_{50}	AP_{25}	AP_{50}	AP_{25}	AP_{50}
VoteNet	59.78	35.77	51.09	31.81	43.68	29.08	40.46	22.23
Ours	60.34	36.80	51.85	32.98	44.66	31.93	41.84	25.04

Table 7: Performance on the Unbalance Problem in SUN RGB-D

A.9 Performance on the Unbalance Problem

To analyze the performance of the proposed model on the imbalance problem, we conduct experiments using all the classes. Note that we conduct the experiments not only on ScanNet V2, but also on the more unbalanced counterparts. We follow the benchmark [1], to create these counterparts: 1) sorting the classes in descending order according to number of samples in each class, then we have $n_i > n_j$ if $i < j$, where n is the number of samples, i and j denote the index of the classes. 2) reducing the number of training samples per class according to an exponential function $n = n_i * u^i$, where $u \in (0, 1)$. The test set remains unchanged. According to the benchmark [1], we define the imbalance factor of a dataset as the number of training samples in the largest class divided by the smallest. Note that we use P as the value of the imbalance factor in ScanNet V2. Additionally, we add another three sets, whose values of imbalance factor are 10P, 25P and 50P for both ScanNet V2. As shown in Table 6, we achieve comparable performance in the original dataset setting. With the imbalance becoming more severe (e.g., 25P, 50P), our approach outperforms the baseline more. Note that our focus is on few-shot 3D object detection, where representation learning of new categories becomes the top consideration of algorithm design. This few-shot problem is more useful for scenarios where many new categories appear frequently and require the system to quickly adapt to recognize them. However, the long-tailed problem focuses on how to learn good representations and classifiers that can deliver good performance for both head and tail categories. We believe that dedicated designs can further improve the performance of long-tailed 3D object detection. We will also add the results and analysis for the long-tailed setting in our paper and hope to inspire more future investigations.

A.10 Limitation Analysis

Although the 3D cues of point clouds are more stable since they can get rid of some visual distractors, such as lighting and perspectives, some factors still impede the model from better generalization. For instance, in 3D scene understanding, if the point cloud in the training set is dense and that of the test set is sparse, a model often performs poorly, which can be treated as a cross-domain problem. Regarding few-shot 3D object detection, the performance might degrade if there is such a large domain gap between base classes and novel classes. Even though the basic geometric features are learned in the base classes, they might not be generalized well to the novel classes due to the difference in point cloud sparsity. The performance of this model has much room for improvement. One way to achieve better performance is large-scale pre-training. Large-scale pre-training enables the model to learn more generic features for transfer learning using limited samples, which benefits the community of 2D few-shot learning (i.e., ImageNet Pre-training). For future works, we might resort to the pre-training models in the 2D domain to facilitate the few-shot generalization on 3D few-shot learning and how these techniques can be combined with our method.

References

[1] Cui, Y., Jia, M., Lin, T.Y., Song, Y., Belongie, S.: Class-balanced loss based on effective number of samples. In: IEEE Conference on Computer Vision and Pattern Recognition (2019)

[2] Liu, Z., Zhang, Z., Cao, Y., Hu, H., Tong, X.: Group-free 3d object detection via transformers. In: IEEE International Conference on Computer Vision (2021)

[3] Misra, I., Girdhar, R., Joulin, A.: An end-to-end transformer model for 3d object detection. In: IEEE International Conference on Computer Vision (2021)

[4] Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. In: IEEE International Conference on Computer Vision (2019)

[5] Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems (2017)

[6] Xie, Q., Lai, Y.K., Wu, J., Wang, Z., Zhang, Y., Xu, K., Wang, J.: Mlcvnet: Multi-level context votenet for 3d object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (2020)

[7] Yan, X., Chen, Z., Xu, A., Wang, X., Liang, X., Lin, L.: Meta r-cnn: Towards general solver for instance-level low-shot learning. In: IEEE International Conference on Computer Vision (2019)